**Тема32. Триггеры**

1. **Создание триггеров**

Операторы, как и хранимые процедуры, выполняются в SQL по мере надобности, но что делать, если нужно выполнить их автоматически?

Например, нужно:

1. проверить правильно ли отформатировано введенное значение;
2. определить критический запас товара на складе, т.е. вычесть заказанный товар из запасов на складе;
3. сохранить копию удаленной строки в архиве.

*То есть при наступлении некоторого события возникает потребность в автоматической обработке данных. Для этого и используют триггеры.*

**Триггер**-это один оператор или группа операторов T-SQL, расположенных блоком между BEGIN и END, которые автоматически выполняются средствами SQL Server в ответ на действие следующих операторов: DELETE, INSERT, UPDATE.

Ни в одном из других операторов триггеры не поддерживаются.

**При создании указывают:**

* однозначное имя триггера,
* имя таблицы с которой связывается триггер,
* действие, на которое должен реагировать триггер, (INSERT, UPDATE, DELETE).

Например

CREATE TRIGGER newproduct\_trigger ON Products

AFTER INSERT

AS

SELECT ‘Product added’;

**AFTER** INSERT означает, что будет запущен после успешного выполнения INSERT.

В итоге на экран будет выведено сообщение: “Product added”, т.е. «Товар добавлен» после каждой строки, введенной в таблицу.

***Чтобы проверить этот триггер***, следует ввести одну или несколько строк в таблицу Products, используя INSERT. После каждой успешной операции ввода строки в таблицу на экране появится упомянутое выше сообщение.

Триггеры определяются по событиям в таблице, причем **на каждое событие в таблице допускается только один триггер.**

Таким образом в каждой таблице может поддерживаться до трех триггеров: по одному на каждое событие, связанное с выполнением вставки, обновления или удаления.

*С несколькими событиями может быть связан единственный триггер, например, если требуется выполнить триггер после вставки, обновления его можно определить, как AFTER INSERT, UPDATE.*

Большинство триггеров относится к типу **AFTER**, т.е. они выполняются при наступлении некоторого события.

Другой тип триггера **INSTEAD OF**.

Если требуется, чтобы строки вообще не удалялись из таблицы, можно создать триггер INSTEAD OF, заменяющий DELETE для конкретной таблицы оператором T-SQL, обновляющим строки таким образом, чтобы сделать их неактивными.

Например, установив соответствующий признак в этих строках.

1. **Удаление триггеров**

Для удаления триггера

DROP TRIGGER newproduct\_trigger;

Триггеры можно обновить ALTER TRIGGER.

Можно сначала удалить, а затем создать снова. Иногда требуется выполнить операторы, не выполняя триггеры, вместо того, чтобы удалять триггеры их можно дезактивировать.

--деактивировать триггер

DISABLE TRIGGER имя триггера

ON имя таблицы;

--активировать триггер

ENABLE TRIGGER имя триггера

ON имя таблицы;

SP\_HELPTRIGGER Products;-список выполняемых триггеров.

Запишем триггер

CREATE TRIGGER neworder\_trigger ON Orders

AFTER INSERT AS

SELECT @@IDENTITY

AS order\_num;

Когда новый заказ сохраняется в таблице Orders, SQL Server формирует новый номер заказа и сохраняет его в столбце order\_num.

А этот триггер просто получает данное значение из функции @@IDENTITY и возвращает его. Таким образом, при вводе каждой строки в таблицу Orders этот триггер всегда возвращает новый номер заказа.

Например

INSERT INTO Orders (order\_date,cust\_id)

VALUES (GetDate (),10001);

Вывод

order\_num

20010

Таблица состоит из трех столбцов. Если столбцы order\_date и cust\_id должны указываться явно, то столбец order\_num автоматически формируется средствами SQL Server, а его содержимое также автоматически возвращается благодаря наличию триггера.

CREATE TRIGGER deleteorder\_trigger

ON Orders

AFTER DELETE

AS

BEGIN

INSERT INTO Orders\_archive (order\_num, order\_date, cust\_id)

SELECT order\_num, order\_date, cust\_id

FROM DELETED;

END;

Выполняется при удалении строк из таблицы Orders. Оператор SELECT используется для сохранения удаляемых строк в архивной таблице Orders\_archive! Для этого нужно иметь таблицу Orders\_archive с такими же столбцами как Orders.

CREATE TRIGGER vendor\_trigger ON Vendors

AFTER INSERT, UPDATE

AS

BEGIN

UPDATE Vendors

SET vend\_state=upper(vend\_state)

WHERE vend\_id IN (SELECT vend\_id FROM INSERTED);

END;

Триггер выполняется после ввода или обновления строк в таблице. Когда строки вводятся или обновляются в Vendors, значение в столбце vend\_state заменяются значениями, возвращаемыми из вызова функции Upper(vend\_state).

1. **Дополнительные сведения о триггерах:**
2. для создания триггеров может потребоваться специальный доступ через систему безопасности (если можно осуществлять вставку, обновление и удаление, то можно и применять триггеры);
3. следует использовать для обеспечения согласованности данных, включая регистр букв, форматирование и т.д. происходит всегда и выполняется прозрачно;
4. пример триггера- журнал аудита и регистрации, можно регистрировать изменения в другой таблице – даже состояния до и после изменений;
5. можно вызывать хранимые процедуры и большинство операторов T-SQL.